OULUN AMMATTIKORKEAKOULU

Jarkko Kotaniemi

MOBIILIROBOTIN NAVIGOINTIMENETELMÄN PARANTAMINEN ODOMETRIAN AVULLA

MOBIILIROBOTIN NAVIGOINTIMENETELMÄN PARANTAMINEN ODOMETRIAN AVULLA

Jarkko Kotaniemi
Opinnäytetyö
Kevät 2018
Sähkö- ja automaatiotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Sähkö- ja automaatiotekniikan tutkinto-ohjelma, automaatiotekniikka

---

Tekijä: Jarkko Kotaniemi
Opinnäytetyön nimi: Mobiilirobotin navigointimenetelmän parantaminen odometrian avulla
Työn ohjaajat: Tapio Heikkilä (VTT), Tero Hietanen (Oamk)
Työn valmistumislukukausi ja -vuosi: Kevät 2018 Sivumäärä: 28

Tässä työssä parannettiin Probot Oy:n valmistamalle mobiilirobottialustalle kehitettyä navigointi- ja ohjausmenetelmää. Robotille kehitettiin anturointia ja odometriaa. Työ toteutettiin kokonaisuudessaan VTT:n tiloissa ja se on osa robottialustalle tehtyä kehitystyötä.

Mobiilirobotin moottorien ohjaus toimii momenttiohjauksella, joka tuottaa virheitä ohjauksessa erinäisten vastusvoimien ja mittausvirheiden vuoksi. Anturoinnin ja säädön avulla voidaan korjata momenttiohjauksen aiheuttamat ongelmat. Odometrian avulla robotti kykenee arvioimaan omaa paikkaansa ympäristössä konenäöllä havaittujen maamerkkien ja seinäpintojen suhteen, vaikka maamerkit ja seinäpinnat eivät ole jatkuvasti näkökentässä. Robotinohjausohjelmat on kirjoitettu C++-kielellä käyttäen Qt Creator -ohjelmointiympäristöä.

Lopputuloksena saatiin parannettu navigointi- ja ohjausmenetelmä, jonka avulla robotti kykenee seuraamaan seiniä ja tekemään ennalta määrättyjä käännöksiä konenäöllä havaittujen maamerkkien vierestä. Robotti kykenee ajamaan itsenäisesti ja sen ei tarvitse jatkuvasti nähdä maamerkkejä tai seinäpintoja löytääkseen määrätyt ratapisteet.

Asiasanat: robotiikka, odometria, navigointi

ABSTRACT

Oulu University of Applied Sciences
Degree Programme of Electrical and Automation Engineering, Automation

---

Author: Jarkko Kotaniemi
Title of thesis: Improving the Navigation of a Mobile Robot with Odometry
Supervisors: Tapio Heikkilä (VTT), Tero Hietanen (Oamk)
Term and year when the thesis was submitted: Spring 2018 Number of pages: 28

The goal of this thesis was to improve a navigation and control method made for a mobile robot platform developed by Probot Oy. The entirety of the work was done in the premises of VTT and it is a part of the development of the mobile robot platform.

The driving of the mobile robot's motors works using torque control, which produces errors in the driving due to resistive forces and errors in measurement. Using sensors and controllers, the errors caused by torque control can be fixed. With odometry, the robot is able to estimate its position relative to landmarks and wall surfaces detected by the computer vision even if they are not constantly visible. The programs were written in the C++ language using the Qt Creator development environment.

The final product of this work is an improved navigation and control method that the robot uses to follow walls and to make predetermined turns next to landmarks detected by computer vision. The robot is capable of driving independently and it does not have to see landmarks or wall surfaces in order to find the proper waypoints.

Keywords: robotics, odometry, navigation

ALKULAUSE

Kiitokset VTT:lle ja Oamkin tutkintovastaava Tero Hietaselle loistavasta opinnäytetyöaiheesta. Olen jo pienestä pitäen ollut kiinnostunut robotiikasta, mutta en ole tätä ennen päässyt itse työskentelemään robottien parissa. Opin paljon uutta työn aikana ja pidin hauskaa.

Suuret kiitokset kaikille VTT:llä, jotka olivat minulle apuna. Erityisesti kiitokset työn ohjaajalle VTT:n johtava tutkija Tapio Heikkilälle, jonka asiantuntemus ja tiedonjako olivat iso apu työn tekoon.

Oulussa 20.12.2017
Jarkko Kotaniemi

SANASTO

Odometria	Paikan ja orientaation laskenta antureilta saadun datan perusteella.
C++	C-kielestä kehitetty ohjelmointikieli, johon on lisätty ominaisuuksia kuten oliot
CAN	Automaatioväylä, jota käytetään muun muassa autoissa.
ROS	Robot Operating System on kokoelma kirjastoja ja työkaluja robottien ohjauksen kehitykseen
Qt	Alustariippumaton ohjelmistojen ja graafisten käyttöliittymien kehitysympäristö (lausutaan kuten englannin sana ”cute”).
Python	Monipuolinen tulkattava ohjelmointikieli, joka ei vaadi kääntämistä suorittaakseen
IMU	Inertial Measurement Unit on paketti, joka sisältää antureita kiihtyvyyksien kulmanopeuden ja joskus asentokulmien mittaukseen.

SISÄLLYS

1 JOHDANTO

Työn tarkoituksena oli parantaa Probot Oy:n tekemälle mobiilirobottialustalle tehtyä navigointi- ja ohjausmenetelmää antureiden ja säädön avulla. Työ tehtiin täysin VTT:n tiloissa. Teknologian tutkimuskeskus VTT Oy on kansallisella statuksella toimiva yksi Euroopan johtavista tutkimus- ja teknologiaorganisaatioista (1).

Mobiilirobottialustalle oli tehty aikaisemmin koordinaattiohjaus (2) ja konenäköohjaus (3). Koordinaattiohjauksen oli tehnyt Jussi Pulkkinen ja konenäköohjauksen oli tehnyt Taavi Oksanen. Koordinaattiohjausta ei tässä työssä hyödynnetty, vaan robotin ohjausta parannettiin konenäköohjauksen pohjalta.

Aiemmalla ohjauksella oli jonkin verran ongelmia, joista suurin oli momenttiohjauksen aiheuttama ohjausvirhe. Momenttiohjauksessa mittausvirheet, jotka johtuvat lattiapintojen epätasauksista, renkaiden liukumisesta, vastustavista voimista jne. johtavat ohjausvirheeseen. Tämän lisäksi robotti ei osannut liikkua ilman ympäristöstä havaittuja maamerkkejä tai seinäpintoja, koska robotilla ei ollut sisäistä mallia omasta paikastaan ympäristön suhteen, joten se tarvitsi tietoja ympäristöstään.

Tässä työssä robottiin lisättiin kulmanopeusanturi ja säädin, jonka avulla robotti pystyy pitämään liikkeensä halutussa suunnassa ja halutulla radalla sekä liikkumaan ilman, että konenäkö näyttää mitään. Niin sanottu ”sokea-ajo” on mahdollista, koska anturi pitää säätimen kanssa huolen siitä, että robotti pysyy oikealla radalla.

Robotin konenäkö toimii ROS:n (Robot Operating System) pohjalla ja ohjaus ja säätö toimivat Qt Creatorilla tehdyllä C++-ohjelmalla. Lisäksi robotin moottoreiden ohjaimiin saadaan yhteys Python-koodilla, joka lähettää ohjauskäskyt CAN-muuntimelle, joka taas muuntaa käskyt moottoreille sopivaan muotoon.

2 ROBOTIN OHJAUKSEN MENETELMIÄ

Robotti on kone, joka pystyy suorittamaan monimutkaisia tehtäviä automaattisesti (4). Mobiilirobotti on robotti, joka kykenee liikkumaan omilla keinoillaan. Mobiilirobotin liikkumiskeinot voivat olla monenlaisia, esimerkiksi renkaat, jalat tai propellit. Mobiilirobotteja voidaan käyttää monissa eri käyttötarkoituksissa, joilla on yhteistä se, että robotin pitää kyetä liikkumaan.

2.1 Robotin liikkeen hallinnan anturointi

Täydellisessä maailmassa, jossa olisi mahdollista luoda täydellinen malli ympäristöstä, olisi robotin ohjaus helppoa. Koska tämä ei ole realistisesti mahdollista, robotit tarvitsevat erilaisia antureita, jotta ne saavat takaisinkytkentää ja pystyvät keräämään tietoa ympäristöstään ja reagoimaan ympäristössä oleviin asioihin ja siellä tapahtuviin muutoksiin. Robotit käyttävät yleensä seuraavanlaisia antureita:

- asentoantureita
- raja-antureita
- kosketusantureita
- suunta-antureita
- paikannuslaitteita
- etäisyysantureita
- nopeus- ja liikeantureita
- tunnistuslaitteita.

Asentoanturit ovat antureita, jotka mittaavat kappaleen asentoa suhteessa johonkin toiseen. Niitä voivat olla esimerkiksi kulma-anturit, potentiometrit, resolverit sekä optiset, magneettiset, induktiiviset ja kapasitiiviset anturit. Raja-anturit mittaavat, milloin saavutetaan tai ylitetään jokin raja. Näitä voivat olla esimerkiksi rajakytkimet tai lähestymiskytkimet. Kosketusanturit mittaavat kosketukseen liittyviä voimia. Näitä ovat esimerkiksi kosketusnäytöt sekä voima- ja vääntö-anturit. Suunta-anturit kertovat, mihin suuntaan anturi osoittaa tai miten sen suunta muuttuu. Näitä ovat esimerkiksi kompassit ja gyroskoopit. Paikannuslaitteet kertovat, missä anturi on ympäristön suhteen. Näitä ovat esimerkiksi GPS, radio-, ultraääni- ja heijastinmajakat. Etäisyysanturit kertovat anturin etäisyyden kappaleista tai ympäristöstä. Näitä ovat esimerkiksi kamerat, kaikuluotaimet, laser-etäisyysmittarit

sekä kapasitiiviset ja magneettiset anturit. Nopeus- ja liikeanturit mittaavat anturin nopeutta, kiihtyvyyttä tai liikkumista. Näitä ovat esimerkiksi Doppler-tutkat, kamerat, momenttianturit ja kiihtyvyysanturit. Tunnistuslaitteet osaavat tunnistaa erilaisia kappaleita, tunnisteita tai muita sellaisia. Näitä ovat esimerkiksi kamerat, tutkat tai RFID. (5, s. 144.)

2.2 Robotin liikkeen hallinnan säätö

Robotit tarvitsevat säätöä, koska ilman säätöä robotti ei pysty reagoimaan muuttuvaan tilanteeseen. Ilman säätöä robotti joutuu arvioimaan omia liikkeitä pelkästään sisäisen mallin perusteella. Antureilta saatua tietoa käytetään ympäristön ja robotin havainnointiin ja säädin käyttää hyväksi havaintoja sopiviin ohjausmuutoksiin. Säädin ottaa antureiden mittaustiedon ja vertaa sitä asetusarvoon. Mittauksen ja asetusarvon erotus syötetään säätimeen ja säädin antaa ohjausarvon eteenpäin säätimestä riippuvien toimintojen jälkeen.

Säädintyyppejä on monenlaisia, mutta tämän työn kannalta PID-säädin on olennainen. PID-säätimien etu on se, että niiden ei tarvitse tietää robotista yhtään mitään. Riittää, että säätimen parametrit viritetään sopiviksi. Yksinkertaiselle mobiilirobotille riittää yksinkertainen säädin, mutta monimutkaisemmat robotit voivat tarvita monimutkaisempaa säädintä.

PID-säätimen termeistä P (proportional) tarkoittaa nykyisen tilanteen muutosta hetkellisin keinoin. I (integral) tarkoittaa kasautuvaa pyrkimystä muuttaa tilannetta edellisten muutosten perusteella. D (derivative) on ennakoiva pyrkimys, joka käyttää hyväksi trendejä tulevien tilojen ennakoimiseksi. (5, s. 139.)

2.3 Odometria

Odometria muodostuu kreikan sanoista *hodos* ja *metron*, jotka tarkoittavat matka ja mittaus. Odometriaa käytetään monilla eri aloilla, mutta ajoneuvoihin ja robotiikkaan liittyen se tarkoittaa ajoneuvon toimilaitteista kerätyn tiedon perusteella arvioitua liikettä. Tietojen ja matemaattisten mallien perusteella voidaan arvioida ajoneuvon tai robotin paikan ja asennon muutosta ajan suhteen. Kun

odometriaa käytetään hyväksi paikan muutoksen arviointiin, sitä myös kutsutaan nimellä dead reckoning eli merkintälasku, jota yleisesti käytetään laivojen navigoinnissa merellä. (5, s. 477.)

Matemaattisen mallin ja ohjaustietojen perusteella voitaisiin periaatteessa tietää robotin paikka milloin vain. Ikävä kyllä oikeassa maailmassa syntyy erilaisia mittausvirheitä, kuten virheet robotin fyysisten mittojen selvittämisessä, ohjauskomentojen epävarmuus, moottorinohjaimien tekemät virheet, renkaista johtuvat virheet (kitka, liukuminen jne.). Näiden mittausvirheiden perusteella syntyy jatkuvaa virhettä paikan ja asennon arvioinnissa, koska robotin moottorien ohjaus perustuu momenttiohjaukseen, jossa moottorien nopeus määritetään käyttämällä hyväksi momenttimittausta. Erilaiset virheet aiheuttavat sen, että moottorit pyörittävät renkaita väärällä nopeudella. Tämä ongelma voidaan kuitenkin ratkaista paikan ylläpidolla, joka vaatii jonkin toisen mittauksen merkintälaskun ulkopuolelta.

Visuaalinen odometria on tekniikka, jossa robotin paikkaa ja orientaatiota arvioidaan kameralta saatujen kuvien perusteella. Jos robotin liikehdintä ei ole tasaista tai ympäristö on epätasainen, antureiden mittauksiin syntyy helposti suuriakin virheitä. Visuaalista odometriaa tarvitaan, koska normaali odometria kerryttää virhettä, joten se täytyy korjata käyttämällä ympäristöstä kameralla tehtyjen havaintojen perusteella.

Visuaalista odometriaa käytetään hyväksi monissa robotiikan sovelluksissa, kuten esimerkiksi Marsiin lähetetyissä luotaimissa (6).

3 ROBOTIN KEHITYKSEEN KÄYTETTYJÄ TYÖKALUJA

Tässä luvussa kerrotaan robotin antureista, robotista itsestään sekä siitä, millä tavoin robottia kehitettiin. Robotin ohjelmistot kirjoitettiin kokonaan käyttäen QtCreator ohjelmointiympäristöä. Robotti on Probot Oy:n valmistama mobiilirobottialusta. Lisätty anturi on IMU-yksikkö, josta käytetiin gyroskooppia mittaamaan kulmanopeutta.

3.1 Qt

Qt on alustariippumaton ohjelmistojen ja graafisten käyttöliittymien kehitysympäristö. Qt käyttää pääosin C++-kieltä ja se sisältää laajennuksia kuten signaalit ja slotit, jotka helpottavat tapahtumien käsittelyä.

Qt Creator on ohjelmointiympäristö, joka käyttää hyväksi Qt:ta. Se sisältää koodieditorin, helppokäyttöisen graafisten käyttöliittymien suunnittelutyökalun ja se tukee useita käyttöliittymiä, kääntäjiä, debuggereita ja versionhallintaohjelmistoja. Qt Creator on suunniteltu pääosin C++-kielelle, mutta se tukee myös muita ohjelmointikieliä erinäisten kirjastojen avulla. (7.)

Tässä työssä Qt:n sisältämät ajoitusmekanismit olivat hyödyllisiä säätimen toteutuksessa. Niiden avulla voitiin synkronisoida säätimen säätöväli ja anturin mittausväli. *QTimer*-oliolle voidaan määrittää aikaväli, jolloin se lähettää signaalin *timeout*. Tämä signaali kytketään sitten slotilla esimerkiksi metodiin, joka lukee anturilta saadun tiedon ja päivittää sen muistiin.

Ajoituksen lisäksi Qt sisältää ominaisuuksia sarja- ja verkkoyhteyksien toimintaa varten. Verkkoyhteyttä tarvittiin kameroilta saatujen datagrammien vastaanottoon ja sarjayhteyttä anturin tietojen lukemiseen.

3.2 IMU-yksikkö

IMU (Inertial Measurement Unit) eli inertian mittausyksikkö on laite, joka mittaa kiihtyvyyksiä, kulmanopeuksia ja joskus suuntakulmaa kiihtyvyysmittareiden, gyroskooppien ja magnetometrien

avulla. IMU-yksiköitä on yleisessä käytössä lentokoneiden ja laivojen (5, s. 483) sekä avaruusaluksien, kuten sateliittien ja luotaimien ohjauksessa. Mittausdatan perusteella voidaan laskea orientaatio, kääntymisnopeus, vauhti ja paikan muutos.

Iso ongelma IMU-yksiköiden käytössä navigointiin on se, että yksikkö aiheuttaa kerääntyvää virhettä. Koska paikan muutos ja orientaatio lasketaan integroimalla kiihtyvyyttä ja kulmanopeutta, jokainen pieni mittausvirhe kasvaa ajan myötä. Tämä johtaa ”driftiin”, joka on jatkuvasti kasvava ero lasketun paikan ja todellisuudessa olevan paikan välillä. Tämän vuoksi tarvitaan jonkin tapainen seurantajärjestelmä, joka korjaa kasvaneen virheen. (5, s. 484.)

KUVA 1. VTT:n Little Node, jossa on IMU-yksikkö.

Kuvassa 1 on VTT:n Little Node -yksikkö, jonka IMU-yksikkö käyttää anturina LSM9DS0-inertiamoduulia. Moduuli mittaa kolmiulotteisesti kiihtyvyyden, kulmanopeuden ja magneettikentän voimakkuuden (8). Yksikköä voidaan lukea käyttämällä USB-yhteyttä tietokoneeseen. Kun yksikölle lähetetään ”1”, niin se lähettää takaisin mittaustulokset. Anturi mittaa jatkuvasti, mutta se ei lähetä tuloksia ilman kysyntää. Robotti käyttää yksikön mittaustiedoista vain yhden akselin kulmanopeutta.

3.3 VTT:n mobiilirobottialusta

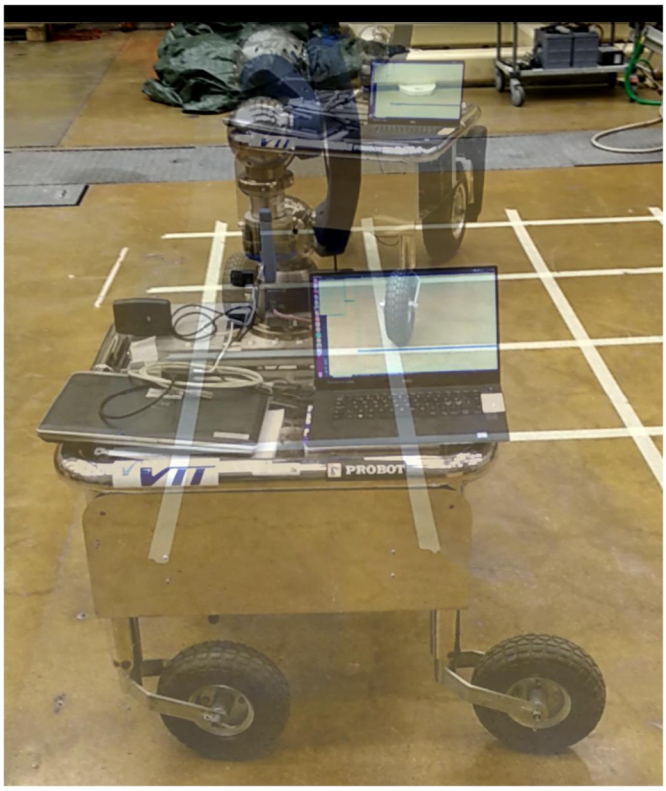
Työssä käytetty mobiilirobottialusta on Probot Oy:n valmistama differentiaaliajoinen mobiilirobotti. Robottiin kuuluu kaksi Mobility Module -yksikköä, jotka toimivat robotin moottoreina. Yksiköissä on

asennonmittaukset, joita ei käytetty hyväksi tässä työssä CAN-muuntimen aiheuttaman virheen vuoksi. Robotti saa yhteyden moottoriyksiköihin Probot Oy:n tekemän Python-koodin avulla. Koodi lähettää ohjausohjelmalta saadut käskyt CAN-muuntimeen, joka taas muuntaa käskyt moottoriyksiköille soveltuvaksi. Qt:llä tehty ohjausohjelma muodostaa TCP/IP -yhteyden Python-moduuliin ja lähettää sille kaksi ohjausarvoa. Yksi arvoista on nimetty ”speed”, joka on skaalattu arvo moottorien nopeudeksi. Toinen arvoista on ”steer”, joka on moottorien nopeusero. Nopeuseron vuoksi moottorit kääntävät robottia vasemmalle tai oikealle.

Mobiilirobotilla on kaksi Kinect-kameraa, joiden avulla robotti pystyy tunnistamaan ympäristöstään maamerkkejä ja seinäpintoja. Kinect-kamerat lähettävät tietokoneelle pistepilven, josta ROS-pohjainen ohjelma tunnistaa maamerkit tai seinäpinnat. Kameroiden ohjausohjelma lähettää tiedot UDP-yhteyden kautta robotin ohjausohjelmalle, joka laskee tietojen perusteella reittipisteet.

Robotin takarenkaat ovat vapaasti pyöriviä kasterimallin renkaita. Renkaat kääntyvät melko helposti, mutta robotin massa aiheuttaa kitkaa akseleissa, jonka vuoksi renkaat eivät aina käänny oikeaan suuntaan. Tämä aiheutti ongelmia moottorien momenttiohjauksessa. Robotti alkaa viettää joko vasemmalle tai oikealle riippuen kumpaan suuntaan robotti oli kääntymässä aikaisemmin, koska takarenkaat jäävät osittain edelliseen asentoon.

KUVA 2. Mobiilirobotti.



KUVA 3. Robotti viettää oikealle yrittäessään kulkea suoraan takarenkaiden vuoksi.

Kuvassa 3 robotti ajaa eteenpäin 2,0 m ja ajautuu sivuun 0,4 m. Näistä muodostuu kulma 11°. Todellisuudessa kulma on jopa suurempi, koska robotin rata on ympyrän kaarella.

Robotin käyttämät Kinect-kamerat ovat käyttökelpoisia, mutta niissä on pieniä ongelmia. Kameroiden näkökenttä on noin 60° vaakasuunnassa ja syvyysnäön maksimietäisyys on 4,5 m (9). Käytännön testien mukaan syvyysnäkö on annettu yläkanttiin. Todellisuudessa kameran näköetäisyys on maksimissaan noin 3 m ja kappaleiden tunnistus onnistuu 0,8-2 m etäisyydellä. Maksimietäisyydellä kameran resoluutio on niin heikko, että kappaleiden tunnistus ei ole enää luotettava.

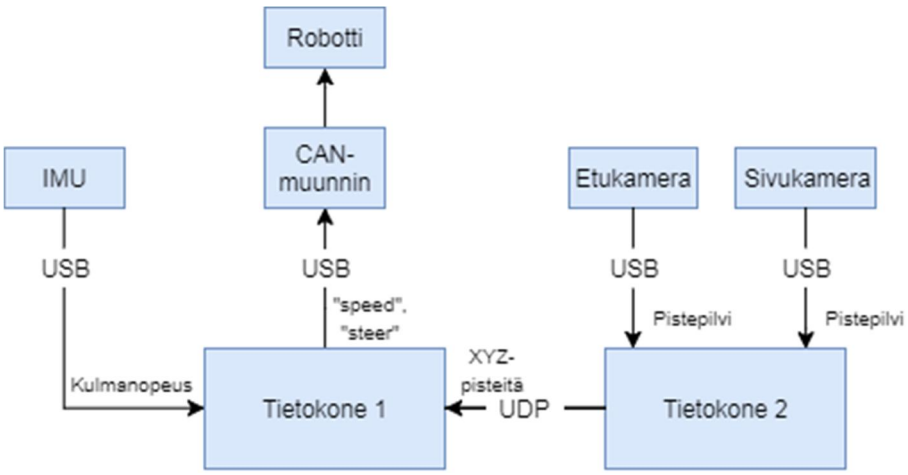
Vaihtoehtona Kinect-kameroille olisi Stereolabsin ZED-stereokamera. ZED-kameran näkökenttä on 110° ja syvyysnäkö on jopa 20 m. Kameran resoluutio voi olla jopa 2.2K. (10.)

4 TYÖN SUORITUS

Työn toteutus alkoi robotin alkuperäisiin ohjelmistoihin ja dokumentaatioihin tutustumisella. Ohjelmistot on kirjoitettu C++-kielellä käyttäen Qt Creator -editoria. Koska C++ ja Qt eivät olleet perusteiden lisäksi tuttuja, täytyi tutustua niihinkin syvällisemmin. Robotin tietoihin tutustumisen lisäksi myös IMU-anturin toiminta piti selvittää. IMU-anturin toiminnan selvittäminen vaati oman ohjelman, jolla saatiin luettua anturin antamat tiedot.

4.1 Robotin ohjauksen tutkiminen ja suunnittelu

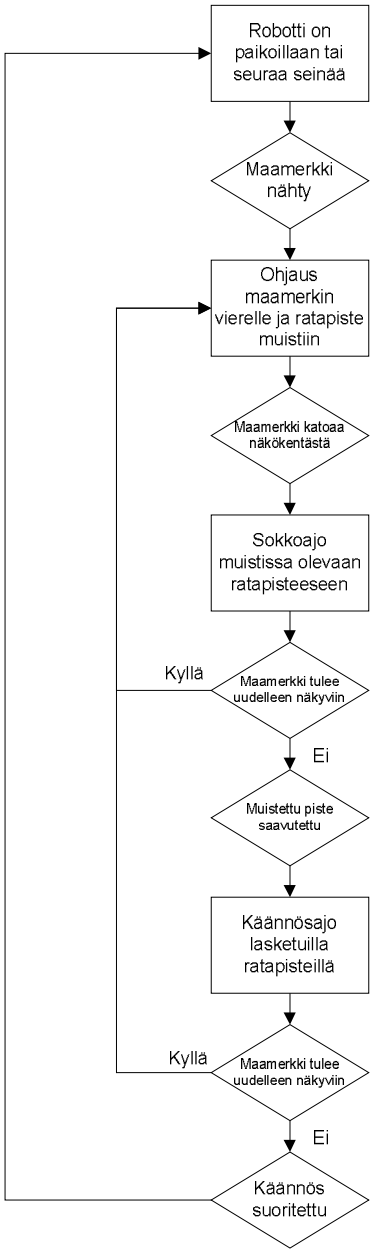
Robotin ohjauksen, anturin ja säädön toiminnan tutkimiseksi tehtiin yksinkertainen ohjelma, johon käyttäjä voi syöttää robotille ohjausarvoja suoraan käyttöliittymän kautta. Alun perin ohjelmaan voitiin syöttää robotille vain ”speed”, ”steer” ja ajoaika. Myöhemmin sitä parannettiin siten, että ohjausarvoja voitiin syöttää sekvensseissä ja ohjausarvojen sijaan syötettiin x- ja y-koordinaatteja, joista x tarkoittaa poikkeamaa robotin keskilinjasta ja y tarkoittaa etäisyyttä robotin origosta. Samaa ohjelmaa oli mahdollista myös käyttää kameraohjauksen simulointiin syöttämällä koordinaatteja sekvenssiin sopivin aikavälein. Tämän ohjelman avulla selvitettiin robotin liikkumiseen liittyvät ominaisuudet, kuten robotin vauhti, kääntösäde ja säätimen parametrit.



KUVA 4. Mobiilirobotin laitteistokaavio.

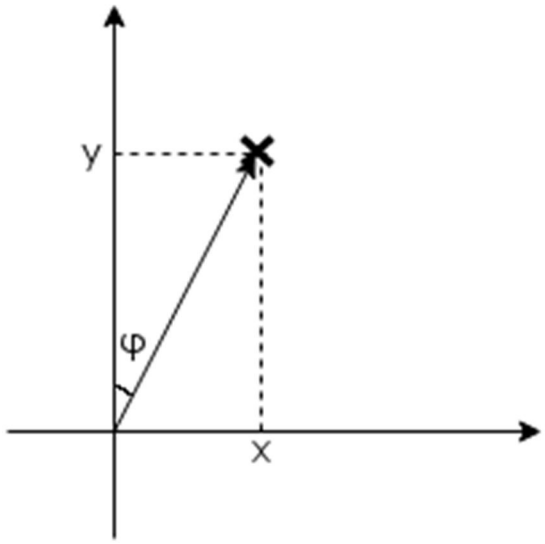
Robotin ohjaus tapahtui ennen yhdellä kannettavalla tietokoneella, mutta Kinect-kamerat lähettävät niin paljon dataa pistepilvissään, että tietokoneen sarjaväylä alkoi mennä tukkoon. Tukos tapahtui,

kun tietokoneeseen yhdistettiin IMU-yksikkö. Tällöin tietokoneessa oli neljä laitetta yhdistettynä sarjaväylään, kaksi Kinect-kameraa, moottorinohjauksen CAN-muunnin ja IMU-yksikkö. Tämä ongelma saatiin ratkaistua lisäämällä toinen tietokone, jolloin yksi koneista keskittyy kameroiden datan selvittämiseen ja toinen keskittyy robotin ohjaukseen (kuva 4). Pistepilvestä segmentoidut seinäpinnat ja maamerkit lähetetään UDP-yhteydellä tietokoneelta toiselle. Tietokone lähettää datagrammit toisen koneen IP-osoitteeseen. Aikaisemmin tietokone lähetti tiedot samalla tavalla, mutta se lähetti ne itselleen local-osoitteeseen. Robotinohjauskone seuraa kaikkia tulevia datagrammeja, mutta koska kameroidenohjauskone on ainut kone verkkoyhteydessä, saapuvat datagrammit voivat olla vain sieltä peräisin.



KUVA 5. Vuokaavio robotin ohjauksesta.

Robotti vastaanottaa ratapisteinä x- ja y-koordinaatteja, joiden perusteella robotti laskee tarvittavan kulmanopeuden, jotta se kulkee ratapisteen läpi liikkuessaan eteenpäin tietyllä vakionopeudella.



KUVA 6. Kulmanopeuden lasku koordinaattien perusteella.

Kameroilta saatujen tietojen perusteella voidaan määrittää robotin ja seuraavan ratapisteen välinen etäisyys *y* ja poikkeama keskilinjasta *x*. Näiden tietojen perusteella lasketaan kulma *φ.*

$$\varphi = \tan^{-1}\frac{x}{y}$$ *KAAVA 1*

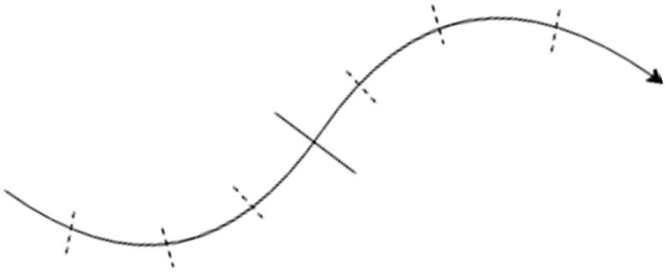
Vaikka ajetut radat ovat osittaisia ympyräratoja, voidaan silti arvioida matka-aikaa ratapisteeseen olettamalla, että aika *t* on sama kuin pelkkään etäisyyteen kuluva aika.

$$t = \frac{y}{v}$$ *KAAVA 2*

Lopulta kulmanopeus *ω* saadaan seuraavasti:

$$\omega = \frac{\varphi}{t} = \frac{\tan^{-1}(x/y)}{y/v} = \frac{v \times \tan^{-1}(x/y)}{y}$$ *KAAVA 3*

Kaava 1 toimii hyvin, kun *x*:n ja *y*:n suhde on pieni. Jos suhde on suuri, kulmanopeus tulee olemaan liian suuri, koska laskettu aika *t* jää liian pieneksi. Tämä ei kuitenkaan haittaa, koska koordinaatit päivittyvät jatkuvasti ja virhe saadaan kompensoitua pois.

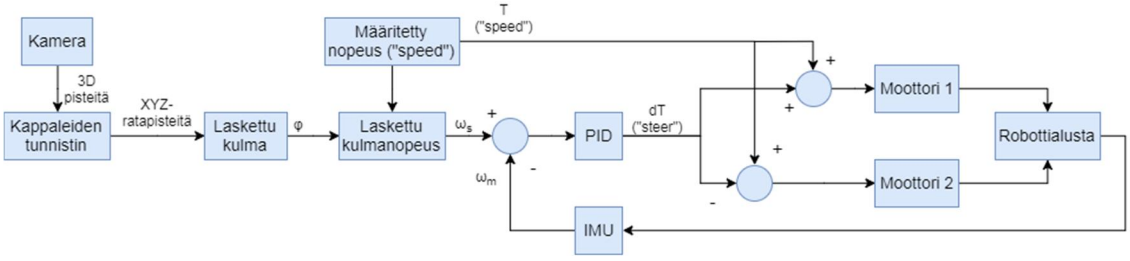


KUVA 7. Robotti kulkee osittaisia ympyräratoja. Katkoviiva kuvaa ohjauskulman säätimen säätöväliä, yhtenäinen viiva kuvaa ratapisteen päivitysväliä. Kuva on liioiteltu.

Robotin liikkeet tapahtuvat aina osittaisina ympyräratoina. Aina kun robotin seuraava ratapiste päivittyy, robotti lähtee kääntymään sitä kohti. Robotti muuttaa kulmanopeuden asetusarvoa vain silloin, kun ratapiste päivittyy. Robotin kameraohjaus päivittää kuvan 400 ms välein ja ohjauskulman säädin toimii 100 ms välein. Näiden perusteella robotin ohjausta ehditään säätää neljä kertaa ratapisteen päivittymisen välillä.

4.2 Ohjauskulman säätö

Robotin rakenteesta johtuvat vastusvoimat aiheuttivat sen, että robotti ei pysynyt halutulla radalla, vaan se alkoi viettää joko vasemmalle tai oikealle riippuen mihin suuntaan robotti oli kääntymässä aikaisemmin. Vastusvoimat voitiin kompensoida pois käyttäen IMU-yksikköä, jolla mitattiin kulmanopeutta, ja kulmanopeuden säädintä. Säätimen säätöväli on sovitettu siten, että säädin säätää robotin ohjausarvoja neljä kertaa ennen kuin kameralta saadaan päivitetty ratapiste.



KUVA 8. Robotin ohjauksen säätökaavio.

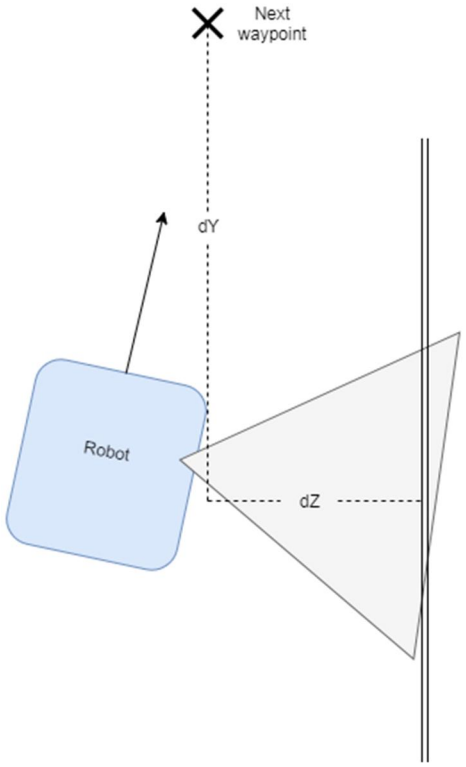
Säätökaaviosta (kuva 6) nähdään, kuinka robotti saa ensiksi kameralta 3D-pisteitä, joista se tunnistaa kappaleita. Kappaleet tulkitaan ja niistä muodostetaan XYZ-ratapisteitä. Ratapisteistä las-

ketaan kulma robotin ja pisteen väliltä ja lasketaan kulmanopeus PID-säädintä varten. Säädin lähettää ”steer” -arvon moottoreille, joista toinen lisää ja toinen vähentää arvot vakioarvosta ”speed”. IMU-yksikkö mittaa robotin kulmanopeutta ja käyttää sitä säätimen mittausarvona.

Robotti käyttää PID-säädintä, koska se on riittävä tällaiseen käyttötarkoitukseen. Momenttiohjauksen virheistä johtuva viettäminen saatiin hyvin korjattua tällä säätimellä. Kuvassa 3 nähdään, kuinka robotin kasterirenkaat aiheuttavat sen, että robotti lähtee viettämään. Säädin käyttää vain P- ja I-termejä säätöön. D-termi aiheuttaa liikaa tärähtelyä ja voi tehdä ohjauksen epästabiiliksi.

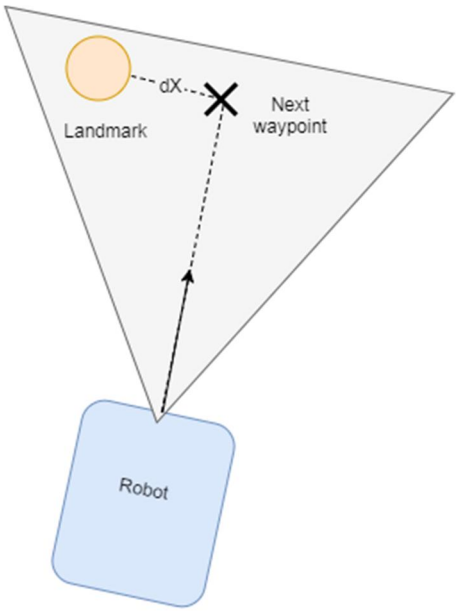
4.3 Kamera-ajo

Kamera-ajossa konenäkö havaitsee ympäristöstä seinäpintoja ja maamerkkejä (3). Seinäpinnoista lasketaan seinäpinnan normaalivektori dZ ja lisätään siihen jonkin verran etäisyyttä dY seinän suuntaisesti, jotta ratapiste tulee sijaitsemaan jonkin matkaa robotin edessä. Kuvassa 9 nähdään esimerkki seinän seuraamisesta.



KUVA 9. Ratapisteiden lasku seinäpintojen perusteella.

Maamerkkien perusteella lasketut ratapisteet ovat yksinkertaisempia. Koska robotti ei voi ajaa maamerkkien läpi, ratapiste annetaan maamerkin vierestä tietyllä etäisyydellä dX. Kuvassa 10 nähdään esimerkki maamerkin mukaisesti määritetystä ratapisteestä.



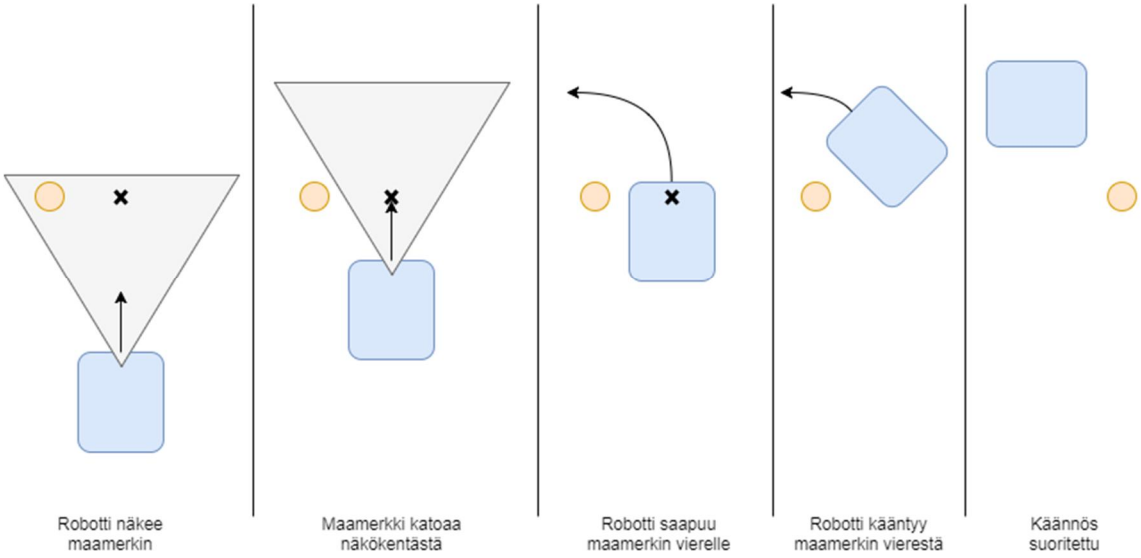
KUVA 10. Ratapisteiden lasku maamerkkien perusteella.

Seinän seuranta on robotille helppoa, koska seinäpinnat ovat suoria. Robotilla menee hetki saada rata suoraksi, mutta radan suoristuttua ohjaus on todella helppoa. Kun x-koordinaatti lähestyy nollaa, ”steer” -arvo myös lähestyy nollaa ja robotin rata suoristuu entisestään. Lopulta ratapisteet ovat suoraan robotin edessä. Ongelmia seinän seurannassa voi tulla, jos seinä ei ole täysin tasainen. Jos seinällä on liian suuri aukko, esimerkiksi avonainen ovi, robotti voi pysähtyä kokonaan tai se voi tulkita aukon seinäpinnaksi, joka on kauempana, jolloin robotti alkaa ohjautua seinää kohti. Tämä voi aiheuttaa liian suuren heilahduksen säätimen asetusarvoissa, jonka vuoksi ohjaus alkaa heittelehtiä. Sama voi tapahtua, jos seinän vierellä on jokin suurehko huonekalu, esimerkiksi kaappi, jonka robotti tulkitsee seinäpinnaksi, joka on lähellä.

Maamerkkien seuranta on myös helppoa. Silloin kun robotti havaitsee maamerkin, robotti ei enää huomioi mitään ympäristössä näkyviä asioita, vaan se keskittyy täysin maamerkkiin. Maamerkit voivat tosin hävitä konenäön näkökentästä, jolloin robotti ei enää voi seurata niitä todellisuudessa. Tämä ei kuitenkaan haittaa, koska robotti muistaa maamerkkien sijainnin ja voi seurata niitä muistin perusteella.

4.4 Sokkoajo

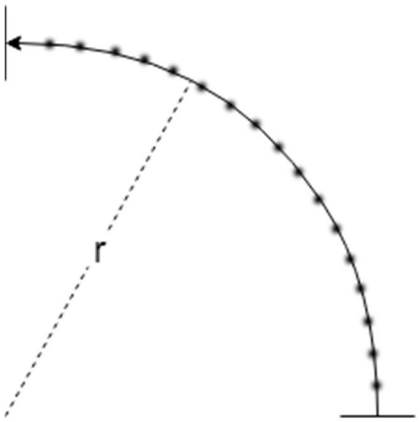
Sokkoajo käynnistyy, kun robotti on nähnyt maamerkin ja se katoaa kameran näkökentästä. Robotti tallentaa muistiin viimeksi lasketun ratapisteen ja ajaa sitä kohti samoin kuin kamera-ajossakin. Sokkoajossa robotti käyttää hyväksi odometriaa, kun se arvioi omaa paikkaansa muistissa olevan pisteen suhteen. Säädön avulla voidaan pitää huoli siitä, että robotti pysyy oikealla radalla. Lisäksi tiedetään, mikä on robotin nopeus. Näiden tietojen perusteella voidaan laskea robotin etäisyys muistetusta pisteestä ja osataan suunnistaa sitä kohti. Jos maamerkki sattuu ilmestymään takaisin konenäön näkökenttään, muistissa oleva ratapiste saadaan siten päivitettyä oikeaksi.



KUVA 11. Tilanne sokkoajosta ja kääntymisestä.

Sokkoajossa robotti ei ole välttämättä kulkemassa suoraan ratapistettä kohti, vaan se mahdollisesti joutuu vielä kääntymään. Jos robotti joutuu kääntymään sokkona, se käyttää hyväksi gyroskoopin kulmanopeutta ja arvioi sen perusteella, milloin robotti on menossa oikeaan suuntaan.

Kääntyminen tapahtuu silloin, kuin robotti saavuttaa maamerkin viereen lasketun ratapisteen. Kääntyminen voi käynnistyä vain silloin, kun robotti on ollut sokkoajossa. Tämä johtuu siitä, että robotti ei voi mitenkään nähdä maamerkkejä konenäöllään, kun ne ovat niin lähellä robottia. Silloin kun robotti kääntyy, se laskee virtuaalisia ratapisteitä sopivalle etäisyydelle ja lopputuloksena robotti tekee 90 asteen käännöksen.



KUVA 12. Käännöksen ratapisteet.

Ratapisteiden määrä riippuu käännöksen säteestä. Mitä suurempi kääntösäde, sitä enemmän tarvitaan ratapisteitä. Kääntösädettä on mahdollista muuttaa erikokoiseksi. Liian tiukalla kääntösäteellä on ongelmana se, että säätimen asetusarvo muuttuu askelmaisesti paljon käännöksen alkaessa. Tästä johtuen robotti ei välttämättä tee oikeanlaista käännöstä, vaan se saattaa tehdä vajaan tai liian suuren käännöksen.

Koska tiedetään, että käännöksen rata on neljännesympyrän kaari (kuva 12), voidaan laskea, kuinka pitkä matka *s* on ja matkassa menevä aika *t*:

$$s = \frac{2\pi r}{4} = \frac{\pi r}{2} \qquad \textit{KAAVA 4}$$

$$t = \frac{s}{v} \qquad \textit{KAAVA 5}$$

$$t = \frac{\pi/_2 r}{v} = \frac{\pi r}{2v} \qquad \textit{KAAVA 6}$$

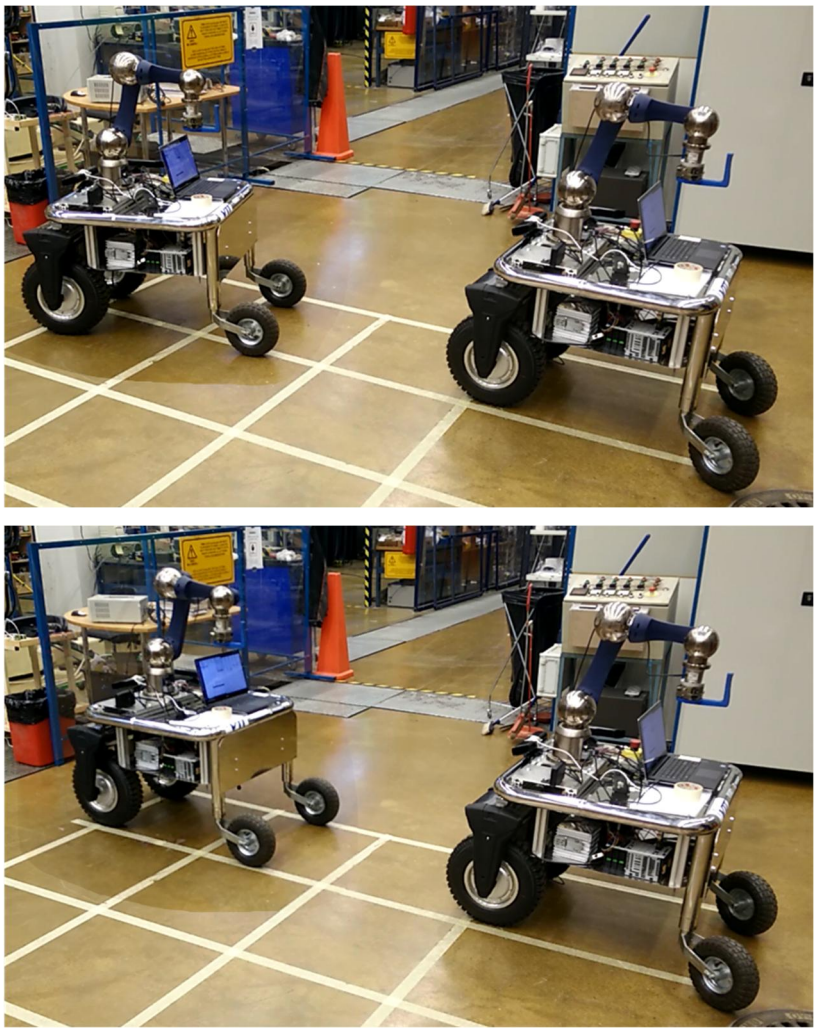
Kun tiedetään radan ajossa menevä aika *t* ja tiedetään ratapisteiden päivitysväli t_s, voidaan laskea käännökselle tarvittava ratapisteiden määrä *n*:

$$n = \frac{t}{t_s} = \frac{\pi r}{2v t_s} \qquad \textit{KAAVA 7}$$

Kun robotti tietää, kuinka monta ratapistettä se tarvitsee käännökseen ja käännös alkaa, robotin seuraava ratapiste määritetään ennalta määrätyn kääntösäteen perusteella. Käännöksessä seuraava ratapiste on aina *x* = *y* = *r*. Laskuri laskee ylöspäin, kunnes se saavuttaa määrättyjen ratapisteiden määrän, jolloin käännös on suoritettu.

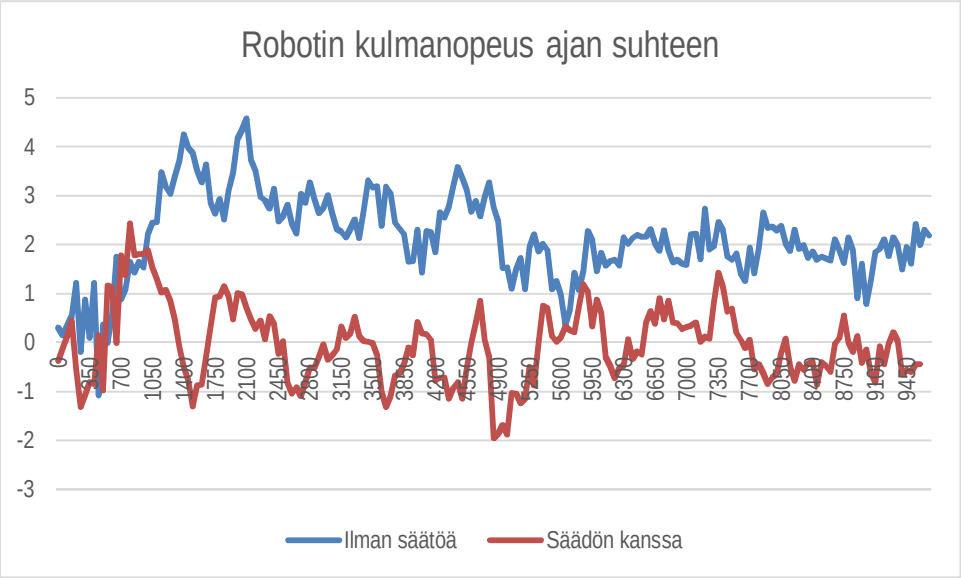
5 TULOSTEN TARKASTELU

Työn edetessä tehtiin monia testejä. Suurin työ oli saada säädin toimimaan, jotta momenttiohjauksen virheet saadaan kompensoitua.



KUVA 13. Säädön avulla korjattu viettäminen. Kaksi tilannetta, joilla oli sama aloitustilanne. Ensimmäisessä säätö on pois ja toisessa säätö on päällä.

Kuvassa 13 nähdään kuinka robotti ajaa vinoon ilman säätöä, vaikka ohjaus käskee robotin ajaa suoraan 2 m. Säädön kanssa robotti taas ei aja enää niin vinoon. Lopputilassa ilman säätöä robotti on noin 39 cm vinossa ja säädön kanssa noin 1 cm.



KUVA 14. Robotin kulmanopeus (deg/s) ajan (ms) suhteen. Robotti ajaa 10 s suoraan.

Kuvassa 14 nähdään robotin kulmanopeus ajan suhteen. Kulmanopeus ilman säätöä on jatkuvasti suurempi kuin nolla, joten robotti on jatkuvasti kääntymässä. Tästä johtuen robotti ajaa vinoon ilman säätöä.

Koska robotin kääntyvät renkaat ovat takana, kääntyessään robotin takaosa liikkuu sivuttain kääntösuuntaa vastaan. Mitä tiukempi kääntösäde, sitä enemmän takaosa liikkuu. Tästä syystä ohjausta muutettiin siten, että käännöksessä maamerkki on sisäkaarteella, jotta robotti ei mitenkään voi osua maamerkkeihin (kuva 15).

KUVA 15. Takarengas käy todella lähellä keilaa.

Lopputestien perusteella mobiilirobotti kykenee ajamaan itsenäisesti. Testiä varten ohjaus muutettiin siten, että robotti kääntyy 180 astetta 90 asteen sijaan. Testissä robotti ajoi 12 kierrosta ennen, kuin testi pysäytettiin.

KUVA 16. Mobiilirobotti ajaa ympyrää itsenäisesti.

6 YHTEENVETO

Työn tavoitteena oli parantaa Probot Oy:n valmistamalle mobiilirobottialustalle tehtyä navigointi- ja ohjausmenetelmää IMU-anturoinnin ja ohjauskulman säädön avulla. Lopputuloksena saatiin parannettu navigointi- ja ohjausmenetelmä, jonka avulla robotti kykenee seuraamaan seiniä ja tekemään ennalta määrättyjä käännöksiä konenäöllä havaittujen maamerkkien vierestä. Robotti kykenee ajamaan itsenäisesti ja sen ei tarvitse jatkuvasti nähdä maamerkkejä löytääkseen määrätyt ratapisteet.

Työ oli tekijälle erittäin kiinnostava ja antoi mahdollisuuden päästä käsiksi robotiikkaan ja sen kehitykseen. Robotiikan lisäksi tekijä sai kokemusta C++-kielen ja Qt Creatorin käytöstä.

Työ on osa mobiilirobottialustalle tehtyä kehitystyötä, joka tulee jatkumaan tulevaisuudessa. Robotin konenäkö toimii tällä hetkellä Kinect-kameroiden avulla, mutta ne korvataan jollain muulla vaihtoehdolla, kuten Stereolabsin ZED-stereokameralla. Jatkossa robottialustan päällä olevaa robottikättä tullaan ottamaan käyttöön.

LÄHTEET

1. Tietoa meistä. 2018. VTT. Saatavissa: http://www.vtt.fi/tietoa-meist%C3%A4. Hakupäivä 29.1.2018.

2. Pulkkinen, Jussi 2016. Mobiilirobotin koordinaattiohjaus. Saatavissa: http://urn.fi/URN:NBN:fi:amk-2016110315720. Hakupäivä 29.1.2018.

3. Oksanen, Taavi 2017. Mobiilirobotin navigointimenetelmä. Saatavissa: http://urn.fi/URN:NBN:fi:amk-2017052910821. Hakupäivä 29.1.2018.

4. Definition of 'robot'. Oxford English Dictionary.

5. Siciliano, Bruno - Khatib, Oussama 2008. Springer Handbook of Robotics. Berlin Heidelberg: Springer-Verlag.

6. Maimone, Mark - Cheng, Yang - Matthies, Larry 2007. Two Years of Visual Odometry on the Mars Exploration Rovers. Journal of Field Robotics 24 (3): 169-186. Saatavissa: https://doi.org/10.1002%2Frob.20184. Hakupäivä 29.1.2018.

7. About Qt. 2018. Qt Wiki. Saatavissa: https://wiki.qt.io/About_Qt. Hakupäivä 29.1.2018.

8. LSM9DS0. 2013. St Datasheet. Saatavissa: https://cdn-shop.adafruit.com/datasheets/LSM9DS0.pdf. Hakupäivä 29.1.2018.

9. Kinect for Windows Sensor Components and Specifications. 2018. Microsoft Developer Network. Saatavilla: https://msdn.microsoft.com/en-us/library/jj131033.aspx. Hakupäivä 29.1.2018.

10. Introduction. 2018. Stereolabs. Saatavissa: https://www.stereolabs.com/documentation/overview/getting-started/introduction.html. Hakupäivä 29.1.2018.